**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | | **01-07-2020** | | | | **Name:** | **Kanaka BS** | |
| **Sem & Sec** | | **6th & A** | | | | **USN:** | **4al17cs039** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | |  | | | | | |
| **Max. Marks** | | | **-** | **Score** | | | **-** | |
| **Pre-placement Training Summary** | | | | | | | | |
| **Topic** |  | | | | | | | |
| **Faculty** |  | | | | **Duration** | | |  |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement**: 2 programs | | | | | | | | |
| **Status: Solved** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **yes** | | | |
| **If yes Repository name** | | | | | <https://github.com/kanakabs/Daily-Status> | | | |
| **Uploaded the report in slack** | | | | | **yes** | | | |

**ONLINE CODING**

**1.  Python Program to Create a Linked List & Display the Elements in the Lists**

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

class LinkedList:

def \_\_init\_\_(self):

self.head = None

self.last\_node = None

def append(self, data):

if self.last\_node is None:

self.head = Node(data)

self.last\_node = self.head

else:

self.last\_node.next = Node(data)

self.last\_node = self.last\_node.next

def display(self):

current = self.head

while current is not None:

print(current.data, end = ' ')

current = current.next

a\_llist = LinkedList()

n = int(input('Enter the elements would you like to add: '))

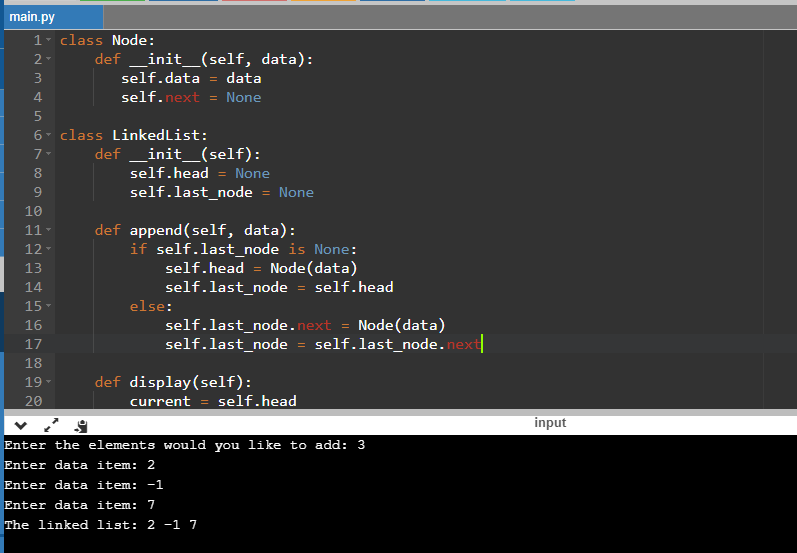
for i in range(n):

data = int(input('Enter data item: '))

a\_llist.append(data)

print('The linked list: ', end = '')

a\_llist.display()



**2. Write a program to find given two trees are mirror or not.**

class Node

{

int data;

Node left, right;

public Node(int data)

{

this.data = data;

left = right = null;

}

}

public class BinaryTree

{

Node a, b;

booleanareMirror(Node a, Node b)

{

if (a == null && b == null)

return true;

if (a == null || b == null)

return false;

returna.data == b.data

&&areMirror(a.left, b.right)

&&areMirror(a.right, b.left);

}

public static void main(String[] args)

{

BinaryTree tree = new BinaryTree();

Node a = new Node(1);

Node b = new Node(1);

a.left = new Node(2);

a.right = new Node(3);

a.left.left = new Node(4);

a.left.right = new Node(5);

b.left = new Node(3);

b.right = new Node(2);

b.right.left = new Node(5);

b.right.right = new Node(4);

if (tree.areMirror(a, b) == true)

System.out.println("Yes");

else

System.out.println("No");

}

}

